Electronic Version
Stylesheet Version v1.1.1

Description

Dynamic PCI-Bus Pre-Fetch with Separate Counters for Commands of Different Data-Transfer Lengths

BACKGROUND OF INVENTION

[0001] This invention relates to burst read accesses, and more particularly to predicting burst-lengths of bus transactions.

[0002] Traditional main memories often employ caches to speed up access times. A subset of the data in the main memory is stored in a cache memory. When the requested data is in the cache, access is more rapid. Each entry in the cache typically contains a portion of its address, such as a tag.

[0003] Various prediction mechanisms exist for pre-fetching data into the cache. For example, when the data is program code, the next several data items can be pre-fetched into the cache, since memory accesses may occur in a linear sequence of addresses. For more random data accesses, pre-fetching may be less effective.

[0004] Such prediction methods used for cache memories are less useful for input-output (IO) bus transactions. IO buses often connect to a variety of peripheral devices. Some devices may transfer a small amount of data while other devices transfer large blocks of data at once. Traditional cache-memory prediction methods fail under these non-homogenous conditions.

[0005] One popular IO bus is the Peripheral Component Interconnect (PCI), used in many personal computers (PCs), computer servers, storage and network systems. The PCI bus can connect to a wide variety of devices, including memory, disk drives, graphics systems, and controllers to other buses, such as Universal-Serial-Bus (USB) and FireWire (IEEE 1394). A wide variety input-output devices can be accessed through these other buses, such as memory cards, pointing devices (mice), music devices, printers, etc. The types of bus accesses for this wide variety of devices is quite varied. Predicting ahead in such a varied environment is challenging, yet a good prediction scheme could improve bus performance.

[0006] What is desired is a prediction method for an input-output bus that connects to a wide variety of devices. A bus-transaction prediction system is desired for accesses over

a PCI bus.

Brief Description of Drawings

[0007] Figure 1 is a diagram of a system with a PCI bridge.

[0008] Figure 2 shows counters that contain statistics used for predicting prefetch burst-lengths.

[0009] Figures 3A-C are flowcharts for dynamic prefetch control using sets of statistic counters.

Detailed Description

[0010] The present invention relates to an improvement in bus prefetching. The following description is presented to enable one of ordinary skill in the art to make and use the invention as provided in the context of a particular application and its requirements. Various modifications to the preferred embodiment will be apparent to those with skill in the art, and the general principles defined herein may be applied to other embodiments. Therefore, the present invention is not intended to be limited to the particular embodiments shown and described, but is to be accorded the widest scope consistent with the principles and novel features herein disclosed.

[0011] Rather than predict addresses of future accesses, the inventor predicts the length of a current bus access. Burst

accesses can have a wide variety of data lengths, from one byte or one word, to hundreds of bytes up to a limit of 4K. The burst-length of earlier accesses is used to predict the burst-length for the current access.

[0012] Records or statistics from previous transactions are stored and used for predicting the current transaction's burst length. For example, statistics can be kept on the burst-length used for prior transactions, and whether the prior predictions were correct, or if the prior prediction predicted too large of a burst length (over-fetch) or too small (under-fetch).

[0013] The past results of prediction are used to adjust the current prediction. For example, when the past prediction was for too large of a burst-length (an over-fetch occurred), then the predicted burst-length for the current access can be reduced. When the past prediction was for too small of a burst-length (an under-fetch occurred), then the predicted burst-length for the current access can be increased.

[0014] Rather than keep prediction statistics for each address, statistics are kept for different types of bus transactions. Since accesses to memory-space addresses tend to have larger data-transfer sizes than accesses to IO-space ad-

dresses, prefetching and prediction is only performed for memory accesses, not for IO reads. Also, only read accesses use prefetching, since write accesses cannot be for lengths greater than the available write data. Thus burst-length prediction is applied only to memory-read accesses.

[0015] PCI bus transactions include a basic memory-read (MR) of one or more bytes, a memory-read-line (MRL) command to read one cache line or more, and a memory-read-multiple (MRM) command to read multiple cache lines. Prediction statistics are kept for each of these 3 read commands. Since the requestor (PCI master) can be on either side of a PCI bridge, statistics are kept for each of the two possible directions, so a total of 6 statistics are kept.

[0016] Figure 1 is a diagram of a system with a PCI bridge. The system could be an advanced personal computer with two PCI buses. Central processing unit (CPU) 10 executes instructions and reads data from memory banks 12. System clock generator 16 generates clocks to various system components, such as to CPU 10, memory banks 12, CPU bridge 14, and PCI-to-PCI bridge 18.

[0017] CPU bridge 14 contains system logic, buffers, state ma-

chines, and other controller logic to allow CPU 10 to read memory banks 12. CPU bridge 14 also contains a PCI-bus controller and buffers to bridge requests from CPU 10 to primary PCI bus 20. A PCI master on primary PCI bus 20 may also request access of memory banks 12 through CPU bridge 14.

[0018] When many devices are connected to primary PCI bus 20, the bus speeds may be degraded due to capacitive loading on the physical lines of the bus. The physical length of the bus may have to be large, adding resistive and capacitive delays of the physical lines themselves. Thus the size of primary PCI bus 20 may be limited, allowing only a few high-speed devices to be attached to primary PCI bus 20.

[0019] Since a PC user may desired to install many peripheral devices or PCI add-on cards to the PC, a second PCI bus is useful. Secondary PCI bus 22 can be larger, or have more available connections, than primary PCI bus 20, allowing for more devices 24 to be installed. PCI devices 24 do not add load to primary PCI bus 20 since PCI-to-PCI bridge 18 buffers the physical lines of secondary PCI bus 22 from the lines of primary PCI bus 20.

[0020] PCI-to-PCI bridge 18 contains bus logic, buffers, state machines, and other controller logic to allow CPU 10 or a

PCI master on primary PCI bus 20 to access devices 24 on secondary PCI bus 22. PCI-to-PCI bridge 18 contains PCI-bus controllers and buffers in both directions, allowing a PCI master on secondary PCI bus 22 to access a PCI device (acting as a slave) on primary PCI bus 20. A PCI master on secondary PCI bus 22 may also request access of memory banks 12 through PCI-to-PCI bridge 18, primary PCI bus 20, and CPU bridge 14.

[0021] PCI-to-PCI bridge 18 contains cache buffer 82, which can store pre-fetched data for the current read command. For example, when CPU 10 reads a memory block on one of devices 24, prefetch controller 62 in PCI-to-PCI bridge 18 may fetch ahead when reading the memory block on device 24. The pre-fetched data is then available for satisfying the current read command if it continues to accept data over and extended burst. The additional data read from device 24 is stored in cache buffer 82. If CPU 10 extends the current cycle with a longer burst access, then the additional data can be sent directly from cache buffer 82 over primary PCI bus 20 to CPU 10 without performing a another read transaction of device 24 over secondary PCI bus 22.

[0022] Data in cache buffer 82 can be organized into cache lines

(lines). The size of the cache line can be fixed or can be determined by a programmable register on PCI-to-PCI bridge 18. For example, the cache line size may be programmed to be 1, 2, 4, 8, 16, or more 32-bit double-words.

[0023] While prefetch controller 62 and cache buffer 82 could be placed in CPU bridge 14, CPU bridge 14 may be part of a larger system-logic chip or set of chips made by another manufacturer. In this embodiment, prefetch controller 62 and cache buffer 82 are used with PCI-to-PCI bridge 18.

[0024] Figure 2 shows counters that contain statistics used for predicting prefetch burst-lengths. Either primary PCI bus 20 or secondary PCI bus 22 may contain the PCI bus master (requestor), while the other bus contains the PCI bus slave, or another bridge to the slave. Separate statistics are kept for each direction. Primary counters 30 contain the statistical counters that are used when the PCI master is on (or connected through) primary PCI bus 20, while secondary counters 32 contain the statistical counters that are used when the PCI master is on (or connected through) secondary PCI bus 22.

[0025] Each of counters 30, 32 are further divided into sets of counters for each type of read operation. Memory-read

counter set 34 contain statistical counters that are used when the PCI master issues a memory-read command that reads one or more bytes of data. Memory-read-line counter set 36 contains statistical counters that are used when the PCI master issues a memory-read-line command to read an entire line of bytes in the cache. Memory-read-multiple counter set 38 contains statistical counters that are used when the PCI master issues a memory-read-multiple command to read more than one line in the cache. Cache lines can vary in size with programmable configurations and hardware chips, but may be 32 double-words (128 bytes) in one embodiment.

[0026] Each of counter set 34, 36, 38 contains several counters. For example, memory-read-line counter set 36 contains completed counter 40, discard counter 42, disconnect counter 44, and prefetch-length counter 46.

[0027] Length counter 46 indicates the predicted length to prefetch. The length can be a number of cache lines to fetch. When a read-memory command is processed, the double-word is read, then the number of cache lines indicated by length counter 46 is pre-fetched into cache buffer 82. The number of cache lines can be 0 to 3 for a 2-bit binary length counter, or 0-7 for a 3-bit binary

length counter.

[0028] Completed counter 40 counts the number of read-memory operations that have completed since counter 40 was last reset by the prefetch controller. Only memory-read operations, not memory-read-line or memory-read-multiple operations increment completed counter 40 in counter set 34.

[0029] Discard counter 42 is incremented for each memory-read operation that is terminated by the PCI bus master, causing data to be discarded from the buffer in PCI-to-PCI bridge 18. Discarding indicates that over-fetching has occurred, since data is thrown away from cache buffer 82.

[0030] Disconnect counter 44 is incremented for each memory-read operation that is terminated by PCI-to-PCI bridge 18 before the PCI master signals the final data transfer. Disconnects occur when the PCI master wants to burst more data but PCI-to-PCI bridge 18 does not have it available in cache buffer 82. Thus disconnects indicate under-fetching.

[0031] Similar statistics are kept for other read operations, and for the other direction of transfers. For example, when the PCI master is on secondary PCI bus 22 and performs a read-memory-multiple operation that is disconnected, the

disconnect counter in the read-memory-multiple set of counters in counters 32 is incremented.

[0032] After the completed counter reaches a pre-defined number, the prefetch-length counter for that type and direction of read-memory operation may be updated. Then the completed counter is cleared. For example, when completed counter 40 for counter set 34 reaches 3, the prefetch controller examines discard counter 42 and disconnect counter 44 in counter set 34 and decides whether to adjust prefetch-length counter 46. When over-fetching has occurred for this type and direction of bus operation, then the number of cache lines to prefetch can be reduced by decrementing prefetch-length counter 46. When under-fetching has occurred for this type and direction of bus operation, then the number of cache lines to prefetch can be increased by incrementing prefetch-length counter 46.

[0033] Figures 3A-C are flowcharts for dynamic prefetch control using sets of statistic counters. In this example, the flow follows memory-read operations that have the PCI master on primary PCI bus 20, which use memory-read counter set 34. Other types or directions of operations can occur between the operations shown, and similar flows occur si-

multaneously for other operations (memory-read-line, memory-read-multiple) and the other direction.

[0034] The set of counters is initialized by clearing the completed counter, discard counter, and disconnect counter in the set, step 50. After the first operation of the type and direction for the counter set (memory-read MR from primary PCI bus 20), completed counter 40 is incremented from 0 to 1, step 52. If the PCI master terminated the operation and caused data to be discarded in cache buffer 82, step 54, then discard counter 42 is incremented from 0 to 1, step 58. When PCI-to-PCI bridge 18 disconnected and terminated the operation while the PCI master wanted more data, step 56, then disconnect counter 44 is incremented from 0 to 1, step 60.

[0035] In Fig. 3B, the second memory-read operation has occurred, so completed counter 40 is again incremented, step 64. When the PCI master terminated the second memory-read operation and caused data to be discarded in cache buffer 82, step 66, then discard counter 42 is incremented, step 70. When PCI-to-PCI bridge 18 disconnected and terminated the second MR operation while the PCI master wanted more data, step 68, then disconnect counter 44 is incremented, step 72.

[0036] When discard counter 42 has already reached 2, step 74, over-fetching has occurred in the last two memory-read operations. To compensate, the amount of pre-fetching for future read-memory operations is reduced by decrementing prefetch-length counter 46. The number of cache lines prefetched for future memory-read operations is reduced, step 78. The procedure returns to the start in Fig. 3A, resetting completed counter 40, discard counter 42, and disconnect counter 44.

[0037] When disconnect counter 44 has already reached 2, step 76, under-fetching has occurred in the last two memory-read operations. To compensate, the amount of pre-fetching for future read-memory operations is increased by incrementing prefetch-length counter 46. The number of cache lines prefetched for future memory-read operations is increased, step 80. The procedure returns to the start in Fig. 3A, resetting completed counter 40, discard counter 42, and disconnect counter 44.

[0038] In Fig 3C, the third memory-read operation has occurred. Completed counter 40 is again incremented, step 84. When the PCI master terminated the third memory-read operation and caused data to be discarded in cache buffer 82, step 86, then discard counter 42 is incremented, step

90. When PCI-to-PCI bridge 18 disconnected and terminated the third MR operation while the PCI master wanted more data, step 88, then disconnect counter 44 is incremented, step 92.

[0039] When discard counter 42 has reached 2, step 94, over-fetching has occurred in the last three memory-read operations. To compensate, the amount of pre-fetching for future read-memory operations is reduced by decrementing prefetch-length counter 46. The number of cache lines prefetched for future memory-read operations is reduced, step 98. The procedure returns to the start in Fig. 3A, resetting completed counter 40, discard counter 42, and disconnect counter 44.

[0040] When disconnect counter 44 has reached 2, step 96, under-fetching has occurred in the last three memory-read operations. To compensate, the amount of pre-fetching for future read-memory operations is increased by incrementing prefetch-length counter 46. The number of cache lines prefetched for future memory-read operations is increased, step 102. The procedure returns to the start in Fig. 3A, resetting completed counter 40, discard counter 42, and disconnect counter 44.

[0041] ALTERNATE EMBODIMENTS

[0042] Several other embodiments are contemplated by the inventors. For example the invention could be applied to other kinds of peripheral buses such as PCI-Express, USB, Firewire, serial AT attachment (SATA), small-computer system interface (SCSI), Ethernet of different speeds, etc. Additional bridges to additional buses could be present with or without the prefetch prediction for transactions across the bridge. More complex bus structures could be used. Multi-port PCI-to-PCI bridges with 3 or more ports can have additional sets of counters for statistics.

[0043] Rather than adjusting the prefetch-length counter when 2 out of a last 3 operation of a certain type and direction show under- or over-fetching, different numbers of operations can be used. For example, adjustment could occur after every 5 operations of a certain type, or when more than 60% of the operations result in discarding or disconnection. The counters could be examined only after the pre-determined number of operations occurs, rather than when the pre-determined threshold number of discards or disconnects occurs. The statistics could be examined periodically rather than after a certain number of completed operations. For example, a timer interrupt could cause the prefetch controller to examine each set of statistic coun-

ters at pre-determined time intervals.

[0044] The prefetch controller could be implemented as hardware logic gates, state machines, software or firmware for a programmable controller or logic array, or some combination. The amount of data for a read command may not be know at first, so accurate prefetch prediction can supply the data from the cache to the bus master directly when prediction is accurate. The cache is invalidated after each read command to prevent stale data. Cache lines can have 64, 128, 256, or more bytes of data. The memory read command may read 1 to 4 bytes or 1 or 8 bytes. The memory-read-line command may read parts of two cache lines rather than just one cache line when the request is not exactly aligned to the cache line.

[0045] Any advantages and benefits described may not apply to all embodiments of the invention. When the word "means" is recited in a claim element, Applicant intends for the claim element to fall under 35 USC Sect. 112, paragraph 6. Often a label of one or more words precedes the word "means". The word or words preceding the word "means" is a label intended to ease referencing of claims elements and is not intended to convey a structural limitation. Such means-plus-function claims are intended to cover not

only the structures described herein for performing the function and their structural equivalents, but also equivalent structures. For example, although a nail and a screw have different structures, they are equivalent structures since they both perform the function of fastening. Claims that do not use the word "means" are not intended to fall under 35 USC Sect. 112, paragraph 6. Signals are typically electronic signals, but may be optical signals such as can be carried over a fiber optic line.

[0046] The foregoing description of the embodiments of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto.